Lecture 3 Review

C++ basic program structure:

- #include<iostream>… ("header files" are essential)
- // comment indicator (stuff to right on <u>same</u> line ignored)
- `std::cout , std::cin` are the std output, input
- `int main ()` (all programs need a main function)

g++ compiler (source code → executable code)

- http://cplusplus.com/doc/tutorial/program_structure.html

!! Scream if you get stuck !!

Number Representation on a Computer

- "Computers are not infinitely precise in their calculations."
- We need to pay attention to significant figures. (As in lab!!)
- Real numbers represented in binary form: fixed-point or floating point
- Fixed point (fixed number of digits before/after decimal point.)
- N bits used to represent number I (e.g., 23.45)

$$I = sign \times (\alpha_n 2^n + \alpha_{n-1} 2^{n-1} + ... \alpha_0 + ... \alpha_{-m} 2^{-m})$$

with $n + m = N - 2$ and N, m , n machine dependent

Advantage: All FxP numbers have same absolute error: 2^{-m-1}

Can represent fractional powers of 2 exactly.

Disadvantage: Cannot represent exactly fractional powers of 10.

➢ We won't use FxP numbers all that much.

Number Representation on a Computer

We will use "floating point numbers:" use a representation of a number where the decimal can float around wrt sig figs and then adjust matters via an exponent. Think scientific notation.

<u>Advantage</u>: Greater range of numbers can be represented wrt FxP rep.

- We'll use floating point rep for numbers almost exclusively.

$$x_{float} = (-1)^s \times 1.f \times 2^{e-bias}$$

s = sign bit. f = mantissa e = "exponent field" bias = 127_{10}

"real" exponent = p = e – bias (always want $e \geq 0$, $\forall p$)

	s	e	f
Bit position	31	30 23	22 0

Assumption: 4 "bytes" = 32 bits used to store number.

Floating Point Representation of a Number

	s	e		f	
Bit position	31	30	23	22	0

$$mantissa = 1.f = 1 + m_{22} \times 2^{-1} + m_{21} \times 2^{-2} + \cdots + m_0 \times 2^{-23}$$

23 bits used to set precision of number (IF 4 bytes used, you decide.)

precision = 1 part in 2^23. What is this in plain English?

Hint: 2^10 = 1024 (call it an even 1000 for estimation purposes).

Q: What the is 2^23? And then 1/2^23 ?

This ratio sets the limit on the precision your computer recognizes, regardless of exponent: e.g., 1.00000005 X 10^-22 = 1.0 X 10^-22

(**IF using 32 bits** to store a number. We will verify on our machines.)

Floating Point Representation of a Number (2)

	s	e		f	
Bit position	31	30	23	22	0

Range of "exponent field e": $0 \le e \le 255$ (Note: 256 values = 2^8.)

Jargon: "normal floating point number": $0 < e < 255$

Q: What is <u>largest</u> <u>positive</u> normal fp number? (Yes, a question to you !)

Recall: $$x_{float} = (-1)^s \times 1.f \times 2^{e-bias}$$

$$mantissa = 1.f = 1 + m_{22} \times 2^{-1} + m_{21} \times 2^{-2} + \cdots + m_0 \times 2^{-23}$$

1.f = 1.1111 1111 1111 1111 1111 111

p = e – bias = e_{10} – 127 (p is the "real" exponent you want)

Answer = ?????

“Double Precision” Numbers

Typically require more precision than just 32 bit representation.

Solution: Use 2 X 32 bits = 64 bit representation. (Who knew?)

Very simple to do in C++ (and other languages). See how soon.

	s	e		f		f (cont.)	
Bit position	63	62	52	51	32	31	0

HW problem: Estimate precision for such “double precision” numbers.

Reference: See CP, sec 2.5 -2.7.

Always use double precision numbers for scientific computing.

"Machine Precision"

```
#include <iostream>

using std::cout; using std::cin; using std::endl;

int main()
{

  float one = 1;
  float eps = 0.02;
  int N;
  cout << " N = " ;
  cin >> N;

  cout << "N = " << N << endl;

  for (int i = 0; i < N; ++i){
    eps = eps/2.;
    one = 1. + eps;

    cout << "one =  " << one << " \t step = " << i << "\t eps = " << eps  << endl;

  }
 return 0;
}
```

New

New

New

New

Even w/ double precision (64 bits), computer precision is <u>not</u> infinite.

$x_c = x(1 \pm e)$ w/ $|e| \leq e_m$.

How to measure e_m?

Execute Machine Precision Code

Edit and compile previous program:

`g++ -o mach_precision mach_precision.cc`

Q: What is N?

Q: What is e_m?

Useful linux trick: Put interactive executable in shell “script.”

#!/bin/tcsh –f   Req’d: says what shell to use, takes options.

Your executable

mach_precision << stuff

Must match

30   input

Req’d magic symbol

stuff

Place in file

Troubles getting your script to run? First, `ls –l` *`your_file`*

Use `cx` to make script file executable. Try `which cx`

Help with C++ Variables and For Loop

My head is exploding.

I need something to read quietly, at my <u>own</u> pace.

http://www.cplusplus.com/doc/tutorial/variables.html

http://www.cplusplus.com/doc/tutorial/control.html

Link also available from PHYS 3340 links page

Summary

- Representation of single & double precision real numbers.
- Either representation has a finite precision.
- Code to determine machine precision for single precision numbers.
- Example of variable declaration (single precision real).
- Example of for loop.
- Simple example of a “here document” in shell scripting.

You should have finished linux tutorial

Don’t suffer in silence. Scream for help!!!